Московский государственный технический университет им. Н.Э. Баумана.

Факультет «Информатика и управление»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

Отчет по рубежному контролю №1 по курсу БКИТ

Выполнил:
студент группы ИУ5-31Б
Бутрим Андрей Александрович

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий Евгеньевич

Подпись и дата:

Подпись и дата:

Москва, 2021 г.

Задание. (Для 3 варианта предметной области и варианта запросов А)
Рубежный контроль представляет собой разработку программы на языке Python, которая выполняет следующие действия:

1) Необходимо создать 2 класса данных в соответствии с Вашим вариантом предметной области, которые связанны отношением один-ко-многим и многие-ко-многим.
2) Необходимо создать списки объектов классов, содержащих тестовые данные (3-5 записей), чтобы первичные и вторичные ключи соответствующих записей были связаны по идентификаторам.
3) Необходимо разработать запросы в соответствии с Вашим вариантом. Запросы сформулированы в терминах классов «Сотрудник» и «Отдел», которые используются в примере. Необходимо перенести эти требования в свой вариант предметной области.

Результатом рубежного контроля является документ в формате PDF, который содержит текст программы и результат выполнения.

Запросы

1) «Водитель» и «Автопарк» связаны соотношением один-ко-многим. Выведите список всех связанных водителей и автопарков, отсортированный по автопаркам, сортировка водителей произвольная.
2) «Водитель» и «Автопарк» связаны соотношением один-ко-многим. Выведите список автопарков с суммарным количеством часов работы всех водителей каждого автопарка, отсортированный по количеству часов.
3) «Водитель» и «Автопарк» связаны соотношением многие-ко-многим. Выведите список всех автопарков, в названии которых есть слово «такси» и список всех их водителей.

```python
from operator import itemgetter


class Dri:
    """Водитель"""
    """Переменная hours - кол-во часов работы в неделю"""
    def __init__(self, id, fio, hours, park_id):
        self.id = id
        self.fio = fio
        self.hours = hours
        self.park_id = park_id


class Park:
    """Автопарк"""

    def __init__(self, id, name):
        self.id = id
        self.name = name


class DriPark:
    def __init__(self, park_id, dri_id):
        self.park_id = park_id
        self.dri_id = dri_id


# Автопарки
parks = [
    Park(1, 'Яндекс такси'),
    Park(2, 'Ситимобил'),
    Park(3, 'Gett такси'),

    Park(11, 'Uber такси'),
    Park(22, 'Автопарк-Гарант'),
    Park(33, 'Калита')
]
```

```python
# Водители
drivers = [
    Dri(1, 'Рыжов', 50, 1),
    Dri(2, 'Спиряков', 25, 2),
    Dri(3, 'Осипов', 15, 3),
    Dri(4, 'Медведев', 35, 3),
    Dri(5, 'Гончаров', 55, 3),
]

drivers_parks = [
    DriPark(1, 1),
    DriPark(2, 2),
    DriPark(3, 3),
    DriPark(3, 4),
    DriPark(3, 5),

    DriPark(11, 1),
    DriPark(22, 2),
    DriPark(33, 3),
    DriPark(33, 4),
    DriPark(33, 5),
]


def main():
    """Основная функция"""

    # Соединение данных один-ко-многим
    one_to_many = [(d.fio, d.hours, p.name)
                   for p in parks
                   for d in drivers
                   if d.park_id == p.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(p.name, dp.park_id, dp.dri_id)
                         for p in parks
                         for dp in drivers_parks
                         if p.id == dp.park_id]

    many_to_many = [(d.fio, d.hours, park_name)
                    for park_name, park_id, dri_id in many_to_many_temp
                    for d in drivers if d.id == dri_id]

    print('Задание А1')
    print("#Список связанных водителей и автопарков, отсортированный по автопаркам")
    res_11 = sorted(one_to_many, key=itemgetter(2))
    print(res_11)

    print('\nЗадание А2')
    print("#Список автопарков и суммарным количеством часов работы их водителей,")
    print("отсортированный по кол-ву часов")
    res_12_unsorted = []
    # Перебираем все парки
    for p in parks:
        # Список водителей парка
        p_drivers = list(filter(lambda i: i[2] == p.name, one_to_many))
        # Если отдел не пустой
        if len(p_drivers) > 0:
            # Зарплаты водителей автопарка
            p_hours = [hours for _, hours, _ in p_drivers]
            # Суммарная зарплата водителей автопарка
            p_hours_sum = sum(p_hours)
            res_12_unsorted.append((p.name, p_hours_sum))

    # Сортировка по суммарной зарплате
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
```

```
    print(res_12)

    print('\nЗадание А3')
    print("#Список всех автопарков, у которых в названии есть 'такси' и список всех их
водителей")
    res_13 = {}
    # Перебираем все автопарки
    for p in parks:
        if 'такси' in p.name:
            # Список водителей автопарка
            p_drivers = list(filter(lambda i: i[2] == p.name, many_to_many))
            # Только ФИО сотрудников
            p_drivers_names = [x for x, _, _ in p_drivers]
            # Добавляем результат в словарь
            # ключ - автопарк, значение - список фамилий
            res_13[p.name] = p_drivers_names

    print(res_13)


if __name__ == '__main__':
    main()
```

Результат выполнения программы:

```
C:\Users\andre\RK>py main.py
Задание А1
#Список связанных водителей и автопарков, отсортированный по автопаркам
[('Осипов', 15, 'Gett такси'), ('Медведев', 35, 'Gett такси'), ('Гончаров', 55, 'Gett такси'), ('Спиряков', 25, 'Ситимобил'), ('Рыжов', 50, 'Яндекс такси')]

Задание А2
#Список автопарков и суммарным количеством часов работы их водителей,
отсортированный по кол-ву часов
[('Gett такси', 105), ('Яндекс такси', 50), ('Ситимобил', 25)]

Задание А3
#Список всех автопарков, у которых в названии есть 'такси' и список всех их водителей
{'Яндекс такси': ['Рыжов'], 'Gett такси': ['Осипов', 'Медведев', 'Гончаров'], 'Uber такси': ['Рыжов']}
```